A tool based on traffic traces and stochastic monotonicity to analyze data centers and their energy consumption

M. Bayati
LACL, Université Paris-Est
Créteil (UPEC), France
mbayati@hotmail.fr

M. Dahmoune
LACL, Université Paris-Est
Créteil (UPEC), France
mdahmoune@yahoo.fr

J.M. Fourneau
DAVID, Université de
Versailles St Quentin, France
jean-
michel.fourneau@uvsq.fr

N. Pekergin
LACL, Université Paris-Est
Créteil (UPEC), France
nihal.pekergin@u-pec.fr

D. Vekris
DAVID, Université de
Versailles St Quentin, France
dimitrios.vekris@uvsq.fr

ABSTRACT

We present a tool to study the trade-off between energy consumption and performance evaluation. The tool uses real traffic traces to model arrivals, and it allows to consider general discrete arrival processes. Some servers are switched on (resp. off) if the monitored QoS becomes less (resp. more) than the *up* (resp. *down*) threshold. A set of threshold couples and the cost function taking into account both the performance measure and the energy consumption are provided by the user. The tool determines the best one for this cost function among the analyzed scenarios. Our method is numerically based but it takes into account some stochastic properties of the model to speed up the computation.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]; G.3 [**Probability and Statistics**]; I.6 [**Simulation and modelling**]

Keywords

Queues, Energy Saving, Discrete Stochastic Process, Numerical Analysis, Data Center

1. INTRODUCTION

Power consumption is one of the main research problems to be considered when designing an efficient cloud computing service [2]. One of the main difficulties is the provisioning and the dimensioning of the service capacities in the presence of a highly fluctuating rate of job arrivals. An over dimensioning of the servers leads to a poor system utilization. The energy consumed by an idle server is about 65% of the energy consumed by an operational server [5]. Hence, one realistic way to reduce significantly the power consumption of a data center is to power down some servers whenever that can be justified by the load conditions [7].

In [8], the problem of managing a service center to keep power consumption low is considered. The goal is to minimize an objective function which takes into account the power consumption and the number of defected customers due to long waiting times. All servers are not always powered up but managed with respect to some predefined thresholds on the number of clients in the system. Some of the servers are defined as reserve servers. If the number of clients becomes greater than u threshold, the reserve servers are powered up. And they are powered down when the number of clients becomes less than d threshold.

In [6] the authors have presented a very detailed model of a data center architecture which is analyzed through simulations. However the assumptions about the traffic are rather simple.

Our model is quite different. A very simple model is considered. However, some measurements of real traffic are used to represent the job arrivals in an accurate manner. Our approach differs from these methods by several points. First it is numerical rather than analytical or simulation based. We can accommodate general arrival processes. Note that the Markovian assumptions (Poisson arrivals, exponential services and switching times) and the infinite buffer capacity are mandatory in [7].

In this paper, the arrival process is assumed to be stationary for short periods of time and change between periods. This allows us to represent for instance hourly variations of the job arrivals. Real traffic traces are used to find the input arrival process (discrete distribution) of the model. A discrete time queue is considered where the time slot is equal to the service time. The traffic traces are sampled with a sampling period equal to this slot duration. Thus, a batch queue is considered with constant service time where the batch arrival process is stationary for short periods of time. The number of servers evolves with the queue performance so with the traffic. The buffer capacity is assumed to be finite.

The system is analyzed for a finite time period T (let's say a day for instance). This time period is divided into sub-intervals where the batch of arrivals are supposed i.i.d.

VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262652

and the number of servers is constant. During such an interval, the system is modeled by a discrete-time Markov chain. At the end of each interval, the arrival process or the number of servers (or both) change. This model is analyzed numerically to compute the transient distributions for the energy consumption and the performance evaluation. Using stochastic monotonicity, it can be proven that the transient distribution reaches the steady-state (it may happen if the time interval is large enough) and the computation is simplified by using a stationary detection as in Sericola's method [11].

As the model is solved numerically, it is much faster and more accurate than stochastic simulation. However, it is not possible to obtain a proof of optimality for a control policy of the number of operational servers. In the tool, threshold based policies have been implemented. First, some performance distributions such as number of losses or queue size are computed. A decision is taken on the number of servers to be switched on or off based on these distributions and some thresholds provided by the modeler. Two thresholds are needed: one to switch on more servers when the performances are too weak (u), and one to switch off some servers when the energy consumption is too large (d). The modeler can combine the performance measures and the energy consumption in a global index. A set of threshold couples is given by the modeler and the analysis is performed for each couple in this set to find the best one among the analyzed scenarios.

The technical part of the paper is organized as follows. Section 2 describes the queuing model and the control for the number of operational servers. Some theoretical results are also presented to simplify the numerical algorithms. The tool description is given in Section 3. Section 4 is devoted to a detailed example and to the corresponding numerical results.

2. MODEL AND ASSUMPTIONS

First the queuing model is given. Then, the threshold-based policy to manage the number of servers to obtain a trade-off between the QoS measure and the energy consumption is described. Finally, the numerical analysis of the model for each time interval during which the arrival process is stationary and the number of servers is constant is explained. Some theoretical results based on the stochastic monotonicity which are used to improve the numerical analysis are presented. The stochastic ordering will be denoted as $\leq_{st}$ while the equality in distribution will be denoted by $=_{st}$.

2.1 Queuing model

A discrete time model is considered. The service time is assumed to be constant. The slot time is equal to the service time for a job. The arrival traces are sampled with a time interval equal to this slot duration. Thus, the queuing model is a batch arrival queue with constant services and finite capacity buffer.

The number of jobs arriving to the data center during the k^{th} slot will be denoted by $A(k)$ which is a discrete distribution. $Q(k)$ (resp. $D(k)$) denotes the distribution of the buffer length (resp. the number of jobs served) during the k^{th} slot. For any distribution X, $X[i]$ is the probability of item i. There are $smax$ identical servers. The service capacity during slot k is denoted by $s(k)$, and $s(0)$ is the initial number of operational servers. The buffer size is denoted by b. The distribution of the performance measures can be computed by induction on k with the following operators:

- δ_v is the Dirac function with $v \in \mathbb{N}$. It is defined by: $\delta_v[i] = 1$ if $i = v$ and $\delta_v[i] = 0$ otherwise.
- $\otimes$ is the convolution of distributions.
- $Y = SUB_v(X)$ is the distribution X translated by constant v. This function corresponds to a subtraction on the random variable. It is defined by: $Y[i] = X[i - v]$ if $i > v > 0$ and $Y[0] = \sum_{i=0}^{v} X[i]$.
- $Y = MIN_b(X)$ is the distribution X bounded by constant b. This function corresponds to a minimum on the random variable. It is defined by: $Y[i] = X[i]$ if $i < b$ and $Y[b] = \sum_{i=b}^{\infty} X[i]$.

As a discrete-time model is considered, the exact sequence of events during a slot have to be described. The arrivals occur first and they are followed immediately by services. The admission is performed per job according to the Tail Drop policy: a job is accepted if there is a place in the buffer, otherwise it is rejected. Thus, the buffer length for $k \geq 1$,

$$Q(k) = MIN_b(SUB_{s(k)}(Q(k-1) \otimes A(k))). \qquad (1)$$

From now it is assumed that $Q(0) = \delta_0$.

Similarly, the distribution of the number of jobs served $D(k)$ can be described as follows:

$$D(k) = MIN_{s(k)}(Q(k-1) \otimes A(k)). \qquad (2)$$

The distribution of the number of the lost jobs during slot k is:

$$L(k) = SUB_{(s(k)+b)}(Q(k-1) \otimes A(k)). \qquad (3)$$

It is assumed that the input arrivals are independent of the current queue state and the past of the arrival process. Under these assumptions, the model of the queue is a time-inhomogeneous Discrete Time Markov Chains.

The problem we have to deal with is related to the nature of the arrival process. Typically, the job arrivals cannot be assumed to be stationary. The data center adapts to the fluctuation of the process by changing the number of operational servers. Such a policy leads to a trade-off between the performance measure and the energy consumption. But, as the number of servers changes with time, the model becomes more complex to analyze.

The main property used in the analysis is the stochastic monotonicity of the queuing model. Such a property allows to derive stochastic bounds. It also helps to prove the convergence of the numerical algorithms used to solve the model.

2.2 Control of the number of servers

The number of servers may vary according to the traffic and performance indices. More precisely, distributions $Q(k)$ and $L(k)$ are monitored and then some decisions are taken according to some threshold values provided by the modeler. Let k be the current time instant and M be the last time instant in which the number of servers changes. More precisely, the following expectations are monitored:

- $\mathbb{E}(Q(k))$, $\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}(A(i))$,
- $\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}(L(i))$, $\frac{1}{k-M+1}\sum_{i=M}^{k}\mathbb{E}(L(i))$,
- $\sum_j r_j Q(k)[j]$, where r_j is an arbitrary function which represents the reward in state j. Note that if $r_j = j$, $\mathbb{E}(Q(k))$ is obtained.

One of these expectations is chosen to compare with the threshold values d and u given by the user. If the chosen expectation is larger than u, the number of operational servers is increased by c_1. If the expectation is smaller than d, the number of operational servers is decreased by c_2. In general $c_1 = c_2$ but it is not mandatory.

The latency to switch on a server is l_1 and the latency to switch off a server is l_2. They are discrete non negative integer values. It is assumed that during a latency period, a new decision cannot be taken.

2.3 Models for energy consumption

The energy consumption takes into account the number of operational servers and the decision to switch on servers:

- e_0 units of energy per time unit when a server is operational but idle.
- e_1 units of energy per time unit when a server is operational and busy.
- e_{on} units needed to switch on a server. Typically e_0 and e_1 are expressed in Joule per time unit while e_{on} is in Joule.

These parameters are given by the modeler. It is assumed that a non operational server does not consume energy. Let i be the number of jobs, the number of busy servers is $\min(i, s(k))$. Similarly the number of idle servers is $(s(k) - i)^+$. Thus, the average energy used at time k by the operational servers is:

$$\sum_{i=0}^{b} Q(k)[i](e_0(s(k)-i)^+ + e_1 \min(i, s(k))).$$

The cost to switch on c_1 servers at time k is:

$$\Psi(k) \times c_1 \times e_{on}$$

where $\Psi(k)$ is a $0-1$ random variable which is equal to 1 if one decides to switch on c_1 servers at time k. By convention, the energy used to switch on a set of servers is consumed at slot k when the decision is taken although the servers become available only at time $k+l_1$ due to the latency. Remark that all switched off servers will be switched on if their number is less than c_1.

The total energy used is the sum of these quantities along the sample path.

2.4 Time interval analysis

Until now, it has been assumed that the arrivals may depend on k to give the most general description of the system. Now a more detailed description of the temporal properties of the arrivals will be given. A weak stationarity of the arrival process is assumed: on short time scale the traffic is stationary, while for longer period it is not. The arrival process in that case is piecewise stationary. Such an assumption is consistent with the night and day evolution of job arrivals observed by long traces. Thus, the approach is based on two steps:

1. The time period T is decomposed into time intervals such that during these time intervals, the traffic is stationary and the number of operational servers is constant. Note that these intervals cannot be computed before the analysis is completed. Indeed the number of operational servers may depend on the traffic and the size of the queues or the number of losses. Thus, the boundary of the time intervals cannot be known before the computation of all the distributions. However, the time instants where the traffic changes are supposed to be exogenous and fixed by the modeler. The main assumption here is that the modeler has an expertise on the statistical analysis of traffic measurements. The time instants at which the number of servers changes have to be computed during the analysis. For each time interval, we must to know the arrival distribution and the number of operational servers.

2. During a time interval, since the arrivals are stationary, i.i.d. and the number of servers is constant, the underlying model is a time-homogeneous DTMC taking values in a totally ordered state space (i.e. $0 \cdots b$). Thus, theoretical results and algorithms related to the stochastic monotonicity can be used [3, 4, 9] to improve the analysis. Without loss of generality, in the following, it is supposed that the considered DTMC models are ergodic.

2.5 Theoretical results

The main property used to speed up the numerical analysis is the stochastic monotonicity of the queuing model. Such a property allows to derive stochastic bounds. It also helps to prove the convergence of the numerical algorithms. Let $\mathcal{G} = \{1, 2, \ldots, n\}$ be a state space endowed with a total order denoted as $\leq$.

Definition 1. Discrete distribution Q_1 is said to be stochastically smaller than Q_2 (denoted as $Q_1 \leq_{st} Q_2$) if

$$\forall i,\ 1 \leq i \leq n,\ \sum_{p=i}^{n} Q_1[p] \leq \sum_{p=i}^{n} Q_2[p]. \tag{4}$$

Definition 2. A DTMC $\{X^{(n)}, n \geq 0\}$ is stochastically monotone if

$$X^{(0)} \leq_{st} X^{(1)} \implies X^{(n)} \leq_{st} X^{(n+1)},\ \forall n.$$

THEOREM 1. *The queuing model is stochastically monotone (see [1]) for a proof).*

Here, these results are used to prove the convergence of the transient distributions to the steady-state. Let g_{min} and g_{max} be two transient distributions of probability representing two realizations of the stochastic process of the queue size with two initial values equal respectively to the Dirac distribution in 0 and b. g_{cur} is the current distribution initialized to the Dirac distribution in 0 at time 0.

Corollary 1. *For all k:*

$$g_{min}^{(k-1)} \leq_{st} g_{min}^{(k)} \leq_{st} g_{cur}^{(k)} \leq_{st} g_{max}^{(k)} \leq_{st} g_{max}^{(k-1)}.$$

Therefore the convergence can be detected by computing the norm of $(g_{min}^{(k)} - g_{max}^{(k)})$ [3].

Assume that in time interval $[t_i, t_{i+1})$, the traffic is stationary and the number of servers is constant. Thus, the transient distribution of a homogeneous Markov chain has to be analyzed during each time interval. By taking into account the monotonicity of the system and the coupling detection algorithm, unnecessary computations are avoided as stated in Corollary 1 and depicted in Figure 1. When ($g_{min}^{(k)}$ and $g_{max}^{(t)}$) become equal, they couple and they will stay equal for the remaining life of the process.

Algorithm 1 Computation with coupling detection

```
 1: $g_{cur} = \delta_0$
 2: for all time intervals $[t_i, t_{i+1})$ do
 3:   $g_{min} = \delta_0$, $g_{max} = \delta_b$
 4:   for all time instant $k$ in this time interval do
 5:     compute new $g_{min}^{(k)}$ from $g_{min}^{(k-1)}$, $g_{max}^{(k)}$ from $g_{max}^{(k-1)}$
        and $g_{cur}^{(k)}$ from $g_{cur}^{(k-1)}$ using Equation 1
 6:     if $g_{min}^{(k)} =_{st} g_{max}^{(k)}$ then
 7:       Jump to $t_{i+1}$ without any new computation of
          the distributions $g_{min}^{(k)}$, $g_{max}^{(k)}$, et $g_{cur}^{(k)}$ which are
          now all equal.
 8:       exit internal loop due to the COUPLING
 9:     end if
10:   end for
11: end for
```

Two cases may occur: one may observe the convergence due to the coupling before the end of the time interval (i.e. t_{i+1}) or the boundary is reached before the occurrence of the coupling. If the first case, we jump to t_{i+1} as soon as we have detected the coupling without any computation of the distribution. Indeed, as the two distributions have coupled, the steady-state distribution is reached thus it is not necessary to continue the numerical analysis until t_{i+1}. Of course, as the traffic is not stationary after the end of the time period, the obtained distribution is not really the steady-state distribution (but we still call it steady-state to explain that the distribution does not change until t_{i+1}). This numerical procedure has strong connections with the stationarity detection heuristic proposed by Sericola in [11] for the efficient computation of reliability. Here, the stationarity is proved by the coupling while it was only numerically checked in [11]. Note that the tool uses the infinite norm to check the equality in distributions mentioned in Instruction 6.

3. THE TOOL

The tool is based on the numerical simulation of Equations 1, 2 and 3 associated with a controller defined in Section 2.2.

Let $\sigma_0 = 0$ and $n \in \mathbb{N}$. Stationary traffic arrivals is considered during interval $[\sigma_{i-1}, \sigma_i)$ for $i \in 1..n$. Thus, the tool takes as input a sequence of instants $\sigma_1, \ldots, \sigma_n = T$ and the corresponding arrival traffic distribution $A_1, \ldots, A_n$. $A_i, i \in 1..n$ refers to the traffic during $[\sigma_{i-1}, \sigma_i)$. The data center is described by its parameters: b, c_1, c_2, l_1, l_2, $s(0)$, s_{max}, e_0, e_1 and e_{on}. The modeler has to provide the QoS measure that will be monitored to control the number of servers (see Section 2.2).

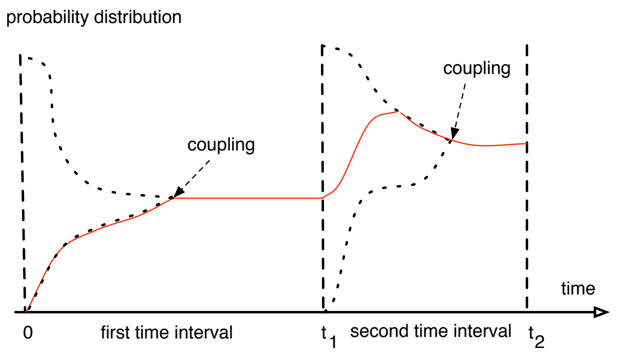

Figure 1: Coupling and stationarity detection. The bounds are plotted in dotted lines while the exact distribution is drawn as a plain red line.

The tool's main function is called for each stationary traffic period $[\sigma_{i-1}, \sigma_i)$. Under the i.i.d. stationary traffic specified by distribution A_i, the function computes the chosen QoS measure for every time slot. For a given slot, if the QoS measure exceeds the upper threshold u then c_1 servers will be switched on after l_1 time units and if the QoS measure becomes less than the lower threshold d then c_2 servers will be switched off after a latency of l_2 time units.

The tool will return the overall energy consumption:

$$\sum_{k=0}^{T} \sum_{j=0}^{b} Q(k)[j](e_0(s(k) - j)^+ + e_1 \min(j, s(k))) + \Psi(k)c_1 e_{on}$$

and the expected number of losses: $\sum_{k=0}^{T} \mathbb{E}(L(k))$

for the period $[0, \sigma_n)$ for every threshold couple (d, u) such that $0 \leq d \leq u \leq b$. Depending on the outputs for the performance and the energy consumption, the modeler can choose the best thresholds for his requirements. Algorithm 2 shows the core control function of the tool in a simplified way. Note that the tool is written in C/C++ and uses the coupling detection algorithm under a sophisticated and optimized implementation.

4. APPLICATION EXAMPLE

This section presents an analysis to find the best thresholds u and d to control the number of servers with respect to the average queue size.

We use the open *clusterdata-2011-2* trace [12, 10], and we focus on the part that contains the job events corresponding to the requests destined to a specific Google data center for the whole month of May 2011. The job events are organized as a table of eight attributes; we only use the column *timestamps* that refer to the arrival times of requests expressed in µ-sec.

We consider frames of one minute to sample the trace and construct two empirical distributions (histograms) one corresponding to arrivals during daytime and the other during nighttime. These distributions have different statistical properties reflecting the fluctuation of traffic between day and night (see Figure 2). For instance, we observe an aver-

Algorithm 2 Tool's main function

```
Require: $n \in \mathbb{N}$, Instants $\sigma_1, \ldots, \sigma_n$
Require: Distributions $A_1, \ldots, A_n$
Require: $b$, $c_1$, $c_2$, $l_1$, $l_2$, $s(0)$, $s_{max}$, $u$, $d \in \mathbb{N}$
Require: $e_0$, $e_1$, $e_{on} \in \mathbb{R}^+$
Ensure: Energy, Loss.
 1: $\sigma_0 \leftarrow 0$, $Q \leftarrow \delta_0$, $s \leftarrow s(0)$
 2: Energy$\leftarrow$ 0, Loss$\leftarrow$ 0
 3: for all time intervals $\sigma_{i-1}, \sigma_i$ do
 4:   if not in latency period then
 5:     if $\mathbb{E}(Q) \geq u$ then
 6:       increase $s$ by $c_1$ after $l_1$ time units
 7:       Energy$\leftarrow$Energy$+c_1 e_{on}$
 8:     end if
 9:     if $\mathbb{E}(Q) < d$ then
10:       decrease $s$ by $c_2$ after $l_2$ time units
11:     end if
12:   end if
13:   $L \leftarrow SUB_{(s+b)}(Q \otimes A_i)$
14:   Loss$\leftarrow$Loss$+\mathbb{E}(L)$
15:   Energy$\leftarrow$Energy$+\sum_{j=0}^{b} Q[j](e_0(s-j)^+ + e_1 \min(j, s))$
16:   $Q \leftarrow MIN_b(SUB_s(Q \otimes A_i))$
17: end for
18: return (Energy, Loss)
```

age of 46 jobs per minute during daytime against an average of 50 jobs per minute during nighttime. However the variance of the daytime traffic is higher than the variance of the nighttime traffic. The tool was used to do the analysis for one day under the following configuration:

n	b	c_1	c_2	l_1	l_2	$s(0)$	s_{max}	e_0	e_1	e_{on}
2	256	1	1	1	0	50	100	13	20	30

The daytime distribution A_1 was considered between $\sigma_0 = 0$ and $\sigma_1 = 720$ minute, and the nighttime distribution A_2 was considered between $\sigma_1 = 720$ and $\sigma_2 = 1440$ minute. The couple (d, u) are defined on the average queue length $(\mathbb{E}(Q(k)))$.

The analysis was done on an *Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz* computer in 49 minutes. Our experimental results showed that the computation becomes 42% faster with the coupling detection.

In Figure 3, for all integer values of u and d, $0 \leq d < u \leq b$ the energy consumption is given. Note that the energy scale is divided by 1000 for the sake of readability.

Figure 4 illustrates losses for all integer values of u and d, $0 \leq d < u \leq b$. It can be seen from these figures that when the energy consumption is high, the number of lost jobs is low and vice versa.

The modeler can combine the loss and the energy in a global index to find the best thresholds.

Note that the tool can be used to analyze the case when the behavior of the system is not stationary along all the considered periods (see Figure 5). For this experience, $b = 1000$, $c_1 = 10$, $l_1 = 1$ and the other parameters are the same as the previous one.

5. CONCLUSION

We advocate that our numerical method, which is based on the stochastic monotonicity of the model, has many advantages compared with simulation and analytical techniques.

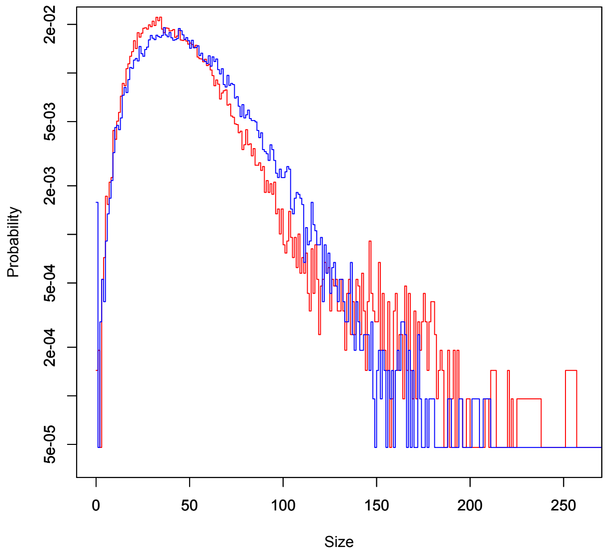

Figure 2: Comparison between daytime (A_1 in red) and nighttime (A_2 in blue) arrival distributions.

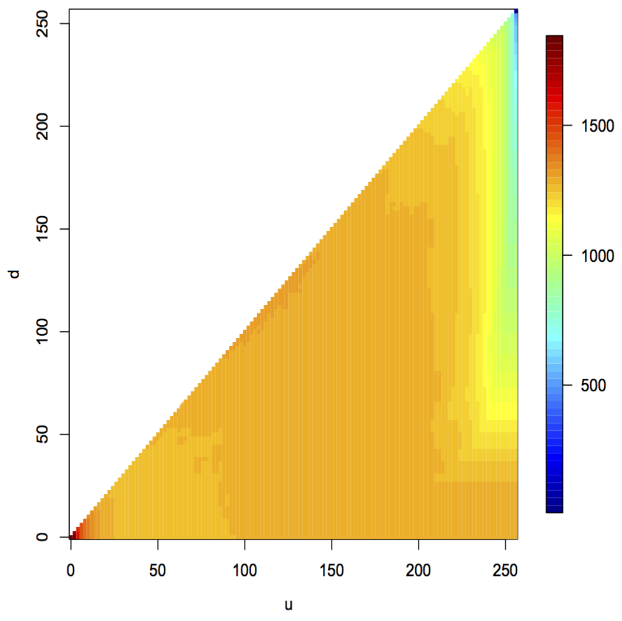

Figure 3: Energy for different values of u and d.

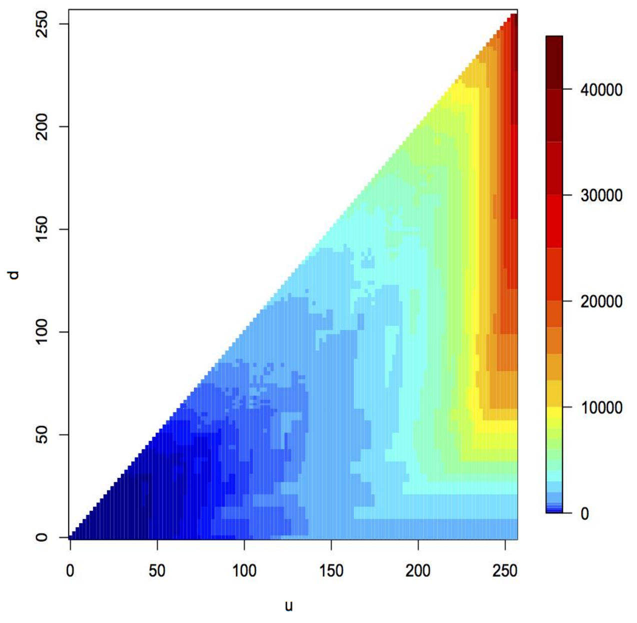

Figure 4: Loss for different values of u and d.

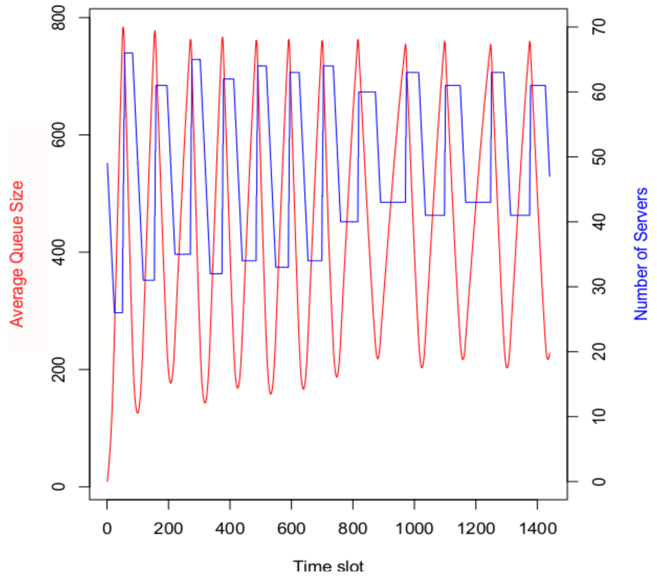

Figure 5: The evolution of the number of operational servers (blue) and the average queue size (red).

We hope that such an idea could be used in many other tools for performance or reliability analysis.

6. ACKNOWLEDGMENTS

The authors are supported by grant ANR MARMOTE: ANR-12-MONU-00019.

7. REFERENCES

[1] F. Aït-Salaht, H. Castel-Taleb, J. Fourneau, and N. Pekergin. Stochastic bounds and histograms for network performance analysis. In *Computer Performance Engineering - 10th EPEW Italy*, volume 8168 of *LNCS*, pages 13–27. Springer, 2013.

[2] A. Berl, E. Gelenbe, M. D. Girolamo, G. Giuliani, H. de Meer, D. M. Quan, and K. Pentikousis. Energy-efficient cloud computing. *Comput. J.*, 53(7):1045–1051, 2010.

[3] A. Busic and J.-M. Fourneau. Monotonicity and performance evaluation: applications to high speed and mobile networks. *Cluster Computing*, 15(4):401–414, 2012.

[4] J.-M. Fourneau and N. Pekergin. An algorithmic approach to stochastic bounds. In *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, volume 2459 of *LNCS*, pages 64–88. Springer, 2002.

[5] A. G. Greenberg, J. R. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *Computer Communication Review*, 39(1):68–73, 2009.

[6] A. Malik, K. Bilal, K. Aziz, D. Kliazovich, N. Ghani, S. Khan, and R. Buyya. Cloudnetsim++: A toolkit for data center simulations in omnet++. In *11th Annual Conference on High-capacity Optical Networks and Emerging/Enabling Technologies (HONET),*, pages 104–108, Dec 2014.

[7] I. Mitrani. Service center trade-offs between customer impatience and power consumption. *Perform. Eval.*, 68(11):1222–1231, 2011.

[8] I. Mitrani. Managing performance and power consumption in a server farm. *Annals OR*, 202(1):121–134, 2013.

[9] A. Muller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks.* Wiley, New York, NY, 2002.

[10] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2012.03.20. Posted at `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`.

[11] B. Sericola. Availability analysis of repairable computer systems and stationarity detection. *IEEE Trans. Computers*, 48(11):1166–1172, 1999.

[12] J. Wilkes. More Google cluster data. Google research blog, Nov. 2011. Posted at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html`.